Journal of Theoretical and Applied Information Technology

15th September 2023. Vol.101. No 17
© 2023 Little Lion Scientific
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

AN EFFICIENCY OF DAWG+ ALGORITHM IN DATA FILTERING FOR DATA INTEGRATION

MIMI HARYANI TASANI[1], MOHD KAMIR YUSOF[2], WAN MOHD AMIR FAZAMIN WAN HAMZAH[3]

[1,2,3] Faculty of Informatics and Computing

Universiti Sultan Zainal Abidin

22200 Besut, Terengganu, Malaysia

Email: [1]mimitasani@gmail, [2]mohdkamir@unisza.edu.my, [3]amirfazamin@unisza.edu.my

ABSTRACT

Data integration is one of the most important components in organization especially for decision maker. The challenge in data integration is to provide a standard view based on different data format, different data schema, etc. A standard view is needed to allow different applications to access the data from different data sources efficiently. One of the issues in providing a standard view is to remove or clean special or unused characters from different data sources. Three (3) different data filtering algorithms have been used in data integration; Knuth Morris Pratt Algorithm (KMP), Boyer-Moore (BM) Algorithm and Backward Non-Deterministic DAWG Algorithm. DAWG algorithm is proven the best of data filtering in term of processing time compared to KMP and BM. In this paper, a modification of the DAWG and named it as a DAWG+ has been proposed. In this algorithm, data will be filtered and removed based on designed a library table. Modification of DAWG algorithm is needs to produce the library table. Three different datasets; SigmodRecord, NASA and DBLP will be used for experiments purposes. Based on the experiments, DAWG+ produced better result in term of data converting response time and query processing response time compared to DAWG. In conclusion, DAWG+ is proven to improve the efficiency during data retrieval process.

Keywords: *Data Integration, Data Filtering, Knuth Morris Pratt, Boyer-Moore, DAWG, Algorithms*

1. INTRODUCTION

Heterogeneous data refers to any data that contains a diverse variety of data types and formats from different data sources [1] The data types include structured data, semi-structured data, and unstructured data. Fully structured data consists of defined data kinds that have styles that facilitate search. Semi structured data forms consist of built - in tags and marks that divide data values, enabling clustering and hierarchy of information. Unstructured data is not like a predefined data model. It can be in text, image, sound, video, or other formats [2]. Databases and documents can consist of semi -structured data types.

Nowadays, most of organization is needed to access data from different data sources. One of approaches can be used to access the data from different data sources is data integration. In data integration, user can access the data using standard data model such as XML, JSON, etc. However, due to missing values, considerable data redundancy, unused characters, and untrustworthiness, they are potentially to reduce efficiency of time performance during access the data [3]. This research is focuses on to remove and clear unused characters which is one the issues in data integration. Data filtering is needed to remove the unused characters in data integration process. Three (3) currently algorithms have been applied in data filtering which are KMP [4], BM [5], and DAWG [6].

This research is focuses on provide a unified view among applications by removing a special or unused characters in filtering process. Three (3) currently algorithms have been applied in data filtering which are KMP [4], BM [5], and DAWG [6].

In this research, we propose to enhance existing DAWG algorithm in data filtering processing for data integration. The purpose of this enhancement is to improve the efficiency during data retrieval process.

Section 1 of this paper discusses the introduction about heterogeneous database, data integration issues and data filtering. This is followed by a review of the data integration and data filtering in Section 2 and Section 3. Section 4 will be discussed about

findings from existing algorithm in data filtering. Section 5 explained about the proposed algorithm. Section 6 discusses the performance of DAWG+ algorithm in data filtering. Meanwhile, conclusion and future work will be discussed in Section 7.

2. DATA INTEGRATION

The practice of merging data from several systems to generate a single collection of information is known as data integration [7]. Its goal is to give users access to a variety of heterogeneous data sources, as well as information about the data's location, storage, and accessibility [8]. Data integration, in technical terms, is the act of merging data from several sources into a single, coherent image. A data integration system's standard design includes the source schema, which is a formal account of how data from sources links to the global view, as well as the mapping, which is a formal account of how data from sources relates to the global view [9].

2.1 XML Data Model

The fact that XML (eXtensible Markup Language) is a self-descriptive format that allows flexible data representation and is an open and free pattern has made it a crucial tool for data representation and exchange over the Web [10]. Data transmission over the internet now largely relies on XML. A very active area of study is updating and retrieving massive amounts of XML data. The XML labelling methods are crucial for handling XML data effectively and securely. As a result, many labelling systems have been put forth [11].

There are more XML data sources now than there were before the Extensible Markup Language (XML) became the standard for data representation and interchange. Because it is text-based and position-independent, XML offers two significant benefits as a language for data representation. Unfortunately, XML is not well suited for data exchange because it is difficult for humans to comprehend [12].

2.2 JSON Data Model

JSON is presently one of the most widely used formats for exchanging data on the Web, but there are very few studies on the subject and no consensus on a theoretical framework for handling JSON. JSON is rapidly rising to the top of the list of the most used Web data exchange formats due to its simplicity and the ease with which it can be read by both humans and machines. This is especially clear when Web services use Application Programming Interfaces (APIs) to communicate with their users because JSON is currently the format of choice for sending API requests and responses over the HTTP protocol [13]. The widely used lightweight semi-structured data format JSON (JavaScript Object Notation) is built on the data types of the JavaScript programming language. In recent years, it has evolved into the primary format for data sharing over the World Wide Web. In the group of database researchers, JSON has grown in popularity. JSON is a semi-structured data format that is commonly used in NoSQL database systems in addition to being compatible with conventional database systems [14].

JSON and XML are two of the most widely used formatting formats for software using web APIs. Each of the formats, JSON and XML has benefits and shortcomings that make them suitable for uses, and each one can be used in accordance with the requirements of the system. The simplicity of JSON's structure, which makes it appropriate for straightforward data transfer, is now widely acknowledged. When compared to JSON, which can only store common data types, one of XML's main benefits is its flexibility, which is demonstrated by the ability to store (theoretically) all possible data types. The trade-off for this flexibility is that the XML format is much more challenging to understand and convert [15].

3. DATA FILTERING

Data filtering is an important tool in system identification and state estimation. A data filtering methodology may be used to discover multi-input and single-output systems based on the maximum probability recursive least squares method. For the input nonlinear system with autoregressive noise, design a multi-innovation adaptive filtering-based stochastic gradient method. They increased convergence and computation efficiency by filtering the input and output signals with a linear filter and splitting the identification model into two sub-models (a noise model, and a filtered system model) [16].

In this section, three data filtering has been reviewed. There are Knuth Morris Pratt Algorithm, Boyer Moore Algorithm, and Backward Non-Deterministic Algorithm.

3.1 Knuth Morris Pratt Algorithm

The KMP algorithm functions by sequentially matching each character's pattern from left to right until one of the criteria is satisfied. Because of this,

the KMP algorithm works well with all different kinds of string queries. Because they can reduce computing time, particularly in big data, search techniques with KMP have been applied in many applications [16].

To discover the specified string positions for text-editing programs, three scientists by the names of Knuth, Morris, and Pratt created the Knuth-Morris-Pratt algorithm. With the help of this algorithm, you can get prefix information about the string or pattern you want to find. In terms of the complexity of the method for pattern matching, the Knuth-Morris-Pratt algorithm differs significantly from brute force. The brute force algorithm, or naive algorithm, matches every possible combination of each character in the text, giving it the difficulty of O (mn). The KMP-Prefix method has an O(m) complexity, where m is the length of the pattern to be searched as a sequence. KMP-Search has an O(n) complexity, where n is the length of the text sequence that serves as the search target [17]. Figure 1 show the process in KMP string search algorithm.

```
Input:
T[0:n-1]
P[0:m-1]
Output:
R[]
New array F[0:m-1]
F[0] ← 1
t ←1
p ← 0
while (t < m - 1) do
        if (P[t] == P[p]) then
                F[t] ← p
                t ← t + 1
        else if (p > 0) then
                p ← F[p]
        else
                F[t] ← 0
                t ←t + 1 c ← 0
while (i < n) do
        if (T[i].== P[j]) then
                i = i + 1
                j = j + 1
                if (j == m) then
                        R[c] = i – m
                        c ← c + 1
                        j = F[m-1]
        else
                j = F[j -1]
                if (j < 0) then
                        i = i + 1j ←0
Return
```

Figure 1: KMP search algorithm

The auxiliary database in this pseudocode is created during pre-processing. Three branches in this step regulate how much t and p increase. T and P grow synchronously by 1 in the first branch. By substituting a lower F[t] for t in the second branch, p t is raised. P increases by 1 in the third branch, but t stays the same. In light of this, either p or the low boundary p t rises. Iteration must stop after 2m + 2 loops. Consequently, O is the temporal complexity.(m). It's easier to start searching. A pattern is aligned or the text pointer is moved one step with each movement. The loop can only run 2n times, according to this truth. In light of this, O is the complexity of the algorithm's search duration.(n).

3.2 Boyer-Moore (BM) Algorithm

Boyer Moore's algorithm finds matches in a sub-linear search period, making it one of the most effective string-matching algorithms currently in use. It does this by merely going left to right through the key string. In the event of a miss, the key string is moved a pre-calculated number of characters to the right until the present character matches. The following letter that hasn't yet been matched is then considered. The number of characters on which the key string match can appear can be calculated since the length of the key string and the position of the current character are known [18].

Boyer-Moore algorithm to the cryptography problem and establishes the viability of doing so to determine the linear complexity of the sequence and the shortest linear shift register, as well as the uniqueness of the shortest linear shift register discovered by the Boyer-Moore algorithm [21].

When there is match string, the Boyer-Moore algorithm follows these stages methodically:

1. The Boyer-Moore method began finding patterns to match at the beginning of the text.
2. This algorithm matches a character-by-character pattern with the corresponding character in the text from right to left, up until one of the following criteria is met:
    - The characters in the pattern and the text do not correspond, for example. (mismatch).
    - If all the characters fit the pattern, the algorithm will alert the user to a discovery in this position.

At the conclusion of the text, the algorithm repeats steps 2 through pattern, shifting pattern to maximize the value of the right-suffix shift and bad-character shift [20].

The BM algorithm contrasts in the window from end to beginning in contrast to the Brute-Force and KMP algorithms. If the last text character to be compared is incorrect and does not follow a pattern, we immediately move the window to bypass the

character. The good suffix rule and the law of bad characters. Figure 2 show bad character rule algorithm.

```
New array Bcr[c][j]
for (c∈ Σ)
        j←0
        while (j<m-1) do
                Value ← −1  /*represents no match*/
                For (i←j-1;i>-1;i--) do
                        If (P[i[==c) then
                                Value←
                                Break  /*exit  for
loop*/
                Bcr[c][j] ←j-value
                J←j+1
```

Figure 2: Bad character rule algorithm

The three instances of the good suffix rule make it more complicated. We align them if the programmed finds a suitable suffix and a portion of the substring is contained in the pattern. Figure 3 show the algorithm to find substring.

```
New array Gsr[0:m-1]
for (i←0;i<m;i++) do
            Gsr[i] ←m
for (i←m-1;i>-1;i--) do
            if (suffix[i]==i+1) then
                        for (j←0;j<m-1-I;j++)
                                    if (Grs[j]==m)then
                                                Grs[j] ←m-1-i
                        for (i←0;i<m;i++) do
                                    Grs[m-1-suffix[i]]  ←m-
1-i
```

Figure 3: Algorithm to find substring

3.3 Backward Non-Deterministic DAW

For the pre-processing and searching segments, respectively, the backward nondeterministic DAWG matching algorithm has a temporal complexity of O(m) and O(n*m) [21]. BND is one of the most famous algorithms to compare strings [22]. The only purpose is to accelerate searching operations because this method makes use of the inherent parallelism of bit operations within a computer word. By first creating a G table in which a bit mask gm...g1 is stored for each and every z character, the Shift-And Algorithm and BDM are used to create BNDM [23].

The Bitap algorithm and the KMP algorithm have a relationship that is very similar to that of the BNDM algorithm and the BDM algorithm, where the Bitap algorithm relies on prefix search while the BNDM algorithm depends on suffix and substring search. The difficulty of BDM's substring-based search technique is in understanding how to conduct such searches. The BNDM algorithm also keeps track of all the locations of substrings that have been discovered in a pattern in a dynamic array. Figure 4 show example of BNDM algorithm.

```
Input:
                        T[0:n-1]
                        P[0:m-1]
Output:
                        R[]
for (c∈Σ) do B[c] ←0
for (i←0;i<m;i++) do
                        B[P[m-1-i]] ←B[P[m-1-i]]+2^i
j←0
while (j+m=≤n) do
                        i←m
                        shift←m
                        D←2^m-1    /*D←1m*/
                                    While D≠0 do
/*T[j+i:j+m] is a substring of pattern*/
                        i←i-1
                        D←D&B[T[j+i]]
                        If (D&2^(m-1) ≠0 then
 /* T[j+i:j+m] is a pattern prefix)*/
                                    If (i==0) then
R.append({j})
/*add j to set R*/
                        Else shift←i
                        D←D<<1
                        j←j+shift
Return R
```

Figure 4: KMP search algorithm

4. MOTIVATION OF THIS RESEARCH

Based on findings in section 3, two different data model can be used as standard unified view in data integration from different data sources. There are XML [10] and JSON [13]. JSON is more versatile in term of structure compared to XML. JSON also can provide fast data transmission compared to XML.

Meanwhile, three different algorithms have been used in data filtering. There are KMP [16], BM [18] and DAWG [23]. The purpose of data filtering is to remove special or unused characters from different data sources. Table 1 show the comparative study of three different algorithm in data filtering.

Based on Table 1, all algorithms are support text, string, and number. KMP is used approach sequentially search, BM is used sub-linear search and DAWG is used substring search. By using substring search, DAWG is producing better performance during searching process compared to KMP and BM.

Table 1: Comparative study of KMP, BM, and DAWG

	Characteristics			
Method.	**Text**	**Number**	**Search**	**Performance**
KMP	/	/	Sequentially	Moderate
BM	/	/	Sub-linear	Moderate
DAWG	/	/	Substring	Fast

Based on findings above, this research is focuses on enhancement of DAWG algorithm and generate JSON as data model. Details about the proposed algorithm will be discussed on Section 5.

5. A PROPOSED OF DAWG+ ALGORITHM

This section will discuss about DAWG+ algorithm. DAWG+ algorithm has been developed based on DAWG algorithm. Figure 5 below shows data filtering model using Backward Non-Deterministic DAWG Matching+ Model (BNDDMN) technique. There main process involves in data filtering; 1) mapping data sources, 2) data filter based on DAWG+ algorithm and 3) data converting. After all process completed, to test the efficiency of DAWG+ algorithm, a few complexity queries will be test in data retrieval process.

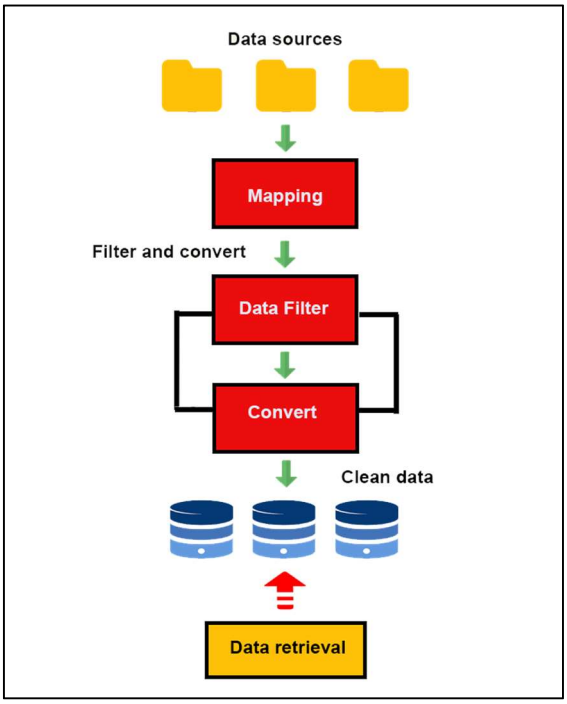

Figure 5: Data filtering process

5.1 Mapping data source

In this process, data sources are selected and mapped for data extraction process.

Definition 1:
Assume S represents a dataset of data sources. S = {ds_1, ds_2, ds_3, ds_n}, where ds_1 until ds_n is an element of S.

Let S consists of data sources ds_1, ds_2, and ds_3. Each data source consists of components which are attributes and elements. These components will be used in extraction process.

Based on Figure 6, dataset of S is mapped to selected sources, where ds_1 until ds_3 is element of S.

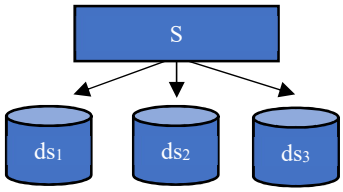

Figure 6: Mapping of data sources

5.2 Data filtering

In this process, each data sources will be filtered to remove special characters based on Table 2.

Table 2: List of special characters

Original special characters	**New characters**
Ä	a
Á	a
ß	b
Ü	u
É	E
É	e
Ó	o
Ö	o
Ç	c
Ñ	n
Ë	e
Õ	o
Í	i

Definition 2:
Assume R represents a dataset of special characters. R = {c_1, c_2, c_3, c_n}, where c_1 until c_n is an element of R.

DAWG+ algorithm will be used to filter and remove special character in dataset R. Figure 7 show

DAWG+ algorithm has been produce based on medication of DAWG algorithm.

```
* j is represented unmatched string
* i is represented number of characters
* D is m, bit of array
* T is prefix of P
* k to j, is read character from left to right
Input:
T [0: n-1]
P [0: m-1]
Process:
L []
for (c∈Σ) do
        B[c]←0
for (i←0; i<m; i++) do
        B[P[m−1−i]] ←B[P[m−1−i]] +2^i
        k ←0
        while (k + m ≤ n) do
                i ← m
                shift ← m
                D←2^m−1
                        while D ≠ 0 do
                        T [k+i: k+m]
                        i ← i–1
                        D←D&B [T [k + i]]
                        if (D&2^(m-1) ≠0) then
                        T [k+i: k+m]
                        if (i==0) then
                           R. append({j})
                        else shift ← i
                           D←D<<1
                           k ← k + shift
Output: L
```

Figure 7: DAWG+ algorithm

According to algorithm in Figure. 8, L is new dataset of S. Figure. 4 show a new dataset of L, after filtering process.

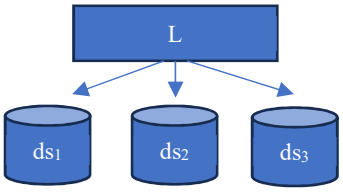

Figure 8: Clean data sources

5.3 Data converting

In this section, new dataset of S will be converted into two (2) different format of data model, XML and JSON. Two algorithms will be used to generate XML and JSON data model. Figure 9 and Figure 10 show the algorithm to generate XML and JSON data model.

Input:
Clean Data Source, L

Output:
XML data model, M, insertion time (t)

Begin:
Read element and attributes from L.
1.1 Identify elements of data source, E.
1.2 Identify attributes of data source, AT.
1.3 Identify value of element, E and attributes, AT.
1.4 Write item = L → E → AT
Repeat Step 1.1 to 1.4 until end of data source, L
Save M to extension of *.xml.
Display data converting response time, t

Figure 9: XML data model

Input:
Clean Data Source, L

Output:
XML data model, M, insertion time (t)

Begin:
Read element and attributes from L.
1.1 Identify elements of data source, E.
1.2 Identify attributes of data source, AT.
1.3 Identify value of element, E and attributes, AT.
1.4 Write item = L → E → AT
Repeat Step 1.1 to 1.4 until end of data source, L.
Save M to extension of *.json.
Display data converting response time, t

Figure 10: JSON data model

5.4 Data retrieval

In this process, a few queries statement will be executed to evaluate the performance in term of data converting response time and query processing response time.

Definition 5:
Assume Q represent dataset of query statement. Q = {q_1, q_2, q_3, q_n} where q_1 until q_n is list of query statement.

Figure 11 show the process of data retrieval from two different data model: XML and JSON. The purpose of this algorithm to retrieve all relevant data from data sources and calculate query processing time based on queries complexity.

Input:
Keyword, Z and query, Q,

Output:
Data retrieved, r, query processing response time, c.

Begin:
Read keyword, Z.
Read query, Q.
Assign Z → Q.
Assign a for start time, b for end time, and c for different.
Search elements and attributes from M
If data is found,
Data retrieved, H = {$data_1$, $data_2$, …, $data_n$}
Else
Display not found.
Repeat until end of data from M.
Calculate, c = b – a.
Display value of c and dataset of H (data retrieved).

Figure 11: Data retrieval algorithm

6. IMPLEMENTATION AND RESULT

This study performed all its experiments on a cloud environment, 1 vCPU with 2GB RAM using an Ubuntu 20.04 (LTS) x64. The software specification for algorithm development is deployed using open-source software, including MySQL version 8.0.31, MySQL community server (GPL) for our database server, Apache/2.4.41 for our web server, PHP as a programming language and phpMyAdmin with administration of MySQL over the Web. The phpMyAdmin (phpMyAdmin: Bringing MySQL to the web) is a free software tool, written in PHP, that supports a wide range of operations on MySQL, MariaDB and Drizzle. The user interface helps one to perform frequently used operations (managing databases, tables, columns, relations, indexes, users, permissions, etc.), with the ability to directly execute any SQL statement. Table 3 exhibits the software configuration.

Table 3: Database configuration

Software	Configuration
Database	MySQL
Web Server	Apache/2.4.41 OpenSSL/1.0.1i PHP/7.4.3 Database client version PHP extension
MySQL	phpMyAdmin
Administration	Version information: 5.2.0, latest stable version: 5.2.0
Programming Language	PHP

In our experiments, we used SigmodRecord, NASA and DBLP as a benchmark dataset. These benchmark datasets have been used for XML approach in experiments purposes in term of data insertion time and query processing response time (Mohsin Marjani et al., 2018). These datasets are download and saved in a *.xml file format. Size of these data are 467KB, 23.9MB and 127.7MB. These datasets provide bibliographical information about computer sciences journals, books, thesis, URL, and proceedings. The overall characteristics of benchmark datasets is tabulated in Table 4. The size in MB represents physical file size, the length represents attributes or labelled as attributes name and the records defines the total number or records.

Table 4: Characteristics of benchmarks datasets

File name	Size (MB)	Length	Record
SigmodRecord	0.467	3737	11526
NASA	23.9	56317	476646
DBLP	127.7	404276	3332130

6.1 Mapping data source

In this section, dataset of S is map to three different sources; SigmodRecord, DBLP and NASA as shows in Figure 12.

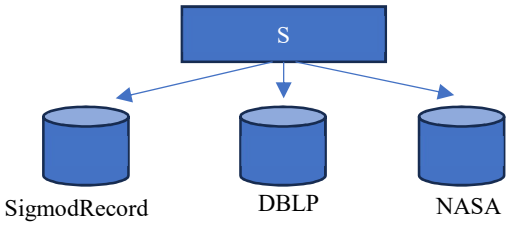

Figure 12: Dataset of S map to SigmodRecord, DBLP and NASA

Figure 13 until Figure 15 shows part of the dataset SigmodRecord, DBLP and NASA. This dataset provides bibliographical information about computer science journals, proceedings, etc.

```
<?xml version="1.0" encoding="utf-8"?>
<SigmodRecord>
 <article>
  <title>Annotated ßibliography on Data Design.</title>
  <initPage>45</initPage>
  <endPage>77</endPage>
  <author>Änthony I. Wasserman</author>
  <author>Änthony I. Wasserman</author>
  <title>Comparison and Mapping of the Relational and CODASYL Data Models - An Ännotated ßibliography.</title>
  <initPage>55</initPage>
  <endPage>68</endPage>
  <author>Gary H. Sockut</author>
  <title>Multisafe - A Data Security Architecture.</title>
  <initPage>26</initPage>
  <endPage>31</endPage>
  <author>Robert P. Trueblood</author>
  <author>Robert P. Trueblood</author>
  <title>A Note on Decompositions of Relational Databases.</title>
:
```

Figure 13: SigmodRecord dataset

```
<?xml version="1.0" encoding="utf-8"?>
<NASA>
 <dataset>
  <title>Proper Motions of Stars in the Zone Catalogue -40 to -52 degrees
of 20843 Stars for 1900</title>
  <altname>1005</altname>
  <altname>1005</altname>
  <altname>1005</altname>
  <identifier>I_5.xml</identifier>
 </dataset>
 <dataset>
  <title>Catalogue of 20554 Faint Stars in the Cape Ästrographic Zone -40 to -52 Degrees
for the Equinox of 1900.0</title>
  <altname>1006</altname>
  <altname>1006</altname>
  <altname>1006</altname>
  <identifier>I_6.xml</identifier>
 </dataset>
:
:
```

Figure 14: NASA dataset

```
<?xml version="1.0" encoding="utf-8"?>
<DBLP>
 <mastersthesis>
  <author>Kurt P. ßrown</author>
  <title>PRPL: A Database Workload Specification Language, v1.3.</title>
  <year>1992</year>
  <school>Univ. of Wisconsin-Madison</school>
 </mastersthesis>
 <mastersthesis>
  <author>Tolga Yurek</author>
  <title>Efficient View Maintenance at Data Warehouses.</title>
  <year>1997</year>
  <school>University of California at Santa Barbara, Department of Computer Science</school>
 </mastersthesis>
  <volume>SRC1997-018</volume>
  <year>1997</year>
  <ee>db/labs/dec/SRC1997-018.html</ee>
  <ee>db/labs/dec/SRC1997-018.html</ee>
  <cdrom>decTR/src1997-018.pdf</cdrom>
 </article>
:
:
```

Figure 15: DBLP dataset

6.2 Data Filtering

In this process, special characters have been found based on Table 1 from three different datasets; SigmodRecord, NASA and DBLP will be eliminated. Figure 16 until Figure 18 shows a new data after elimination process.

```
<?xml version="1.0" encoding="utf-8"?>
  <initPage>33</initPage>
  <endPage>37</endPage>
  <author>Catriel Beeri</author>
  <author>Catriel Beeri</author>
  <title>Actual Conversion Experiences.</title>
  <initPage>20</initPage>
  <endPage>33</endPage>
  <author>James H. Burrows</author>
  <title>Administering a Distributed Data Base Management System.</title>
  <initPage>86</initPage>
  <endPage>99</endPage>
  <author>Henry M. Walker</author>
  <title>Final Report of the ANSI/X3/SPARC DBS-SG Relational Database Task
  Group.</title>
  <initPage>0</initPage>
```

```
    <endPage>62</endPage>
    <author>Michael L. Brodie</author>
    <author>Michael L. Brodie</author>
:
:
```

Figure 16: SigmodRecord

```
<?xml version="1.0" encoding="utf-8"?>
<NASA>
 <dataset>
   <title>Proper Motions of Stars in the Zone Catalogue
-40 to -52 degrees
of 20843 Stars for 1900</title>
   <altname>1005</altname>
   <altname>1005</altname>
   <altname>1005</altname>
   <identifier>I_5.xml</identifier>
 </dataset>
 <dataset>
   <title>Catalogue of 20554 Faint Stars in the Cape
Astrographic Zone -40 to -52 Degrees
for the Equinox of 1900.0</title>
   <altname>1006</altname>
   <altname>1006</altname>
   <altname>1006</altname>
   <identifier>I_6.xml</identifier>
 </dataset>
```

Figure 17: NASA

```
<?xml version="1.0" encoding="utf-8"?>
<DBLP>
 <mastersthesis>
   <author>Kurt P. Brown</author>
   <title>PRPL: A Database Workload Specification
Language, v1.3.</title>
   <year>1992</year>
   <school>Univ. of Wisconsin-Madison</school>
 </mastersthesis>
 <mastersthesis>
   <author>Tolga Yurek</author>
   <title>Efficient View Maintenance at Data
Warehouses.</title>
   <year>1997</year>
   <school>University of California at Santa Barbara,
Department of Computer Science</school>
 </mastersthesis>
   <volume>SRC1997-018</volume>
   <year>1997</year>
   <ee>db/labs/dec/SRC1997-018.html</ee>
   <ee>db/labs/dec/SRC1997-018.html</ee>
   <cdrom>decTR/src1997-018.pdf</cdrom>
 </article>
```

Figure 18: DBLP

6.3 Data Converting

After filtering process is done, dataset of SigmodRecord, DBLP and NASA will be converted into two different data model: XML and JSON. Figure 19 until Figure 21 shows datasets in XML format.

```
<?xml version="1.0" encoding="utf-8"?>
   <initPage>33</initPage>
   <endPage>37</endPage>
   <author>Catriel Beeri</author>
   <author>Catriel Beeri</author>
   <title>Actual Conversion Experiences.</title>
   <initPage>20</initPage>
   <endPage>33</endPage>
   <author>James H. Burrows</author>
   <title>Administering a Distributed Data Base
Management System.</title>
   <initPage>86</initPage>
   <endPage>99</endPage>
   <author>Henry M. Walker</author>
   <title>Final Report of the ANSI/X3/SPARC DBS-
SG Relational Database Task
   Group.</title>
   <initPage>0</initPage>
   <endPage>62</endPage>
   <author>Michael L. Brodie</author>
   <author>Michael L. Brodie</author>
:
:
```

Figure 19: SigmodRecord (.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<NASA>
 <dataset>
   <title>Proper Motions of Stars in the Zone
Catalogue -40 to -52 degrees
of 20843 Stars for 1900</title>
   <altname>1005</altname>
   <altname>1005</altname>
   <altname>1005</altname>
   <identifier>I_5.xml</identifier>
 </dataset>
 <dataset>
   <title>Catalogue of 20554 Faint Stars in the Cape
Astrographic Zone -40 to -52 Degrees
for the Equinox of 1900.0</title>
   <altname>1006</altname>
   <altname>1006</altname>
   <altname>1006</altname>
   <identifier>I_6.xml</identifier>
 </dataset>
```

Figure 20: NASA (.xml)

© 2023 Little Lion Scientific
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

```
<?xml version="1.0" encoding="utf-8"?>
<DBLP>
 <mastersthesis>
  <author>Kurt P. Brown</author>
  <title>PRPL: A Database Workload Specification Language, v1.3.</title>
  <year>1992</year>
  <school>Univ. of Wisconsin-Madison</school>
 </mastersthesis>
 <mastersthesis>
  <author>Tolga Yurek</author>
  <title>Efficient View Maintenance at Data Warehouses.</title>
  <year>1997</year>
  <school>University of California at Santa Barbara, Department of Computer Science</school>
 </mastersthesis>
  <volume>SRC1997-018</volume>
  <year>1997</year>
  <ee>db/labs/dec/SRC1997-018.html</ee>
  <ee>db/labs/dec/SRC1997-018.html</ee>
  <cdrom>decTR/src1997-018.pdf</cdrom>
 </article>
:
:
```

Figure 21: DBLP (.xml)

Meanwhile, Figure 22 until Figure 24 shows datasets in JSON format.


```
[{"attr":"title","data_value":"Annotated Bibliography on Data Design."},{"attr":"initPage","data_value":"45"},{"attr":"endPage","data_value":"77"},{"attr":"author","data_value":"Anthony I. Wasserman"},{"attr":"author","data_value":"Anthony I. Wasserman"},{"attr":"title","data_value":"Architecture of Future Data Base Systems."},{"attr":"initPage","data_value":"30"},{"attr":"endPage","data_value":"44"},{"attr":"author","data_value":"Lawrence A. Rowe"},{"attr":"author","data_value":"Lawrence A. Rowe"},{"attr":"title","data_value":"Database Directions III Workshop Review."},{"attr":"initPage","data_value":"8"},{"attr":"endPage","data_value":"8"},{"attr":"author","data_value":"Tom Cook"},{"attr":"initPage","data_value":"9"},{"attr":"endPage","data_value":"29"
:
:
```


Figure 22: SigmodRecord (.json)


```
[{"attr":"title","data_value":"Proper Motions of Stars in the Zone Catalogue -40 to -52 degrees\nof 20843 Stars for 1900"},{"attr":"altname","data_value":"1005"},{"attr":"altname","data_value":"1005"},{"attr":"altname","data_value":"1005"},{"attr":"identifier","data_value":"I_5.xml"},{"attr":"title","data_value":"Catalogue of 20554 Faint Stars in the Cape Astrographic Zone -40 to -52 Degrees\nfor the Equinox of 1900.0"},{"attr":"altname","data_value":"1006"},{"attr":"altname","data_value":"1006"},{"attr":"altname","data_value":"1006"},{"attr":"identifier","data_value":"I_6.xml"},{"attr":"title","data_value":"Proper Motions of 1160 Late-Type Stars"},{"attr":"altname","data_value":"1014"},{"attr":"altname","data_value":"1014"},{"attr":"altname","data_value":"1014"},{"attr":"identifier","data_value":"I_14.xml"},{"attr":"title","data_value":"Katalog von 3356 Schwachen Sternen fuer das Aequinoktium 1950\n+89:
:
```


Figure 23: NASA (.json)


```
[{"attr":"title","data_value":"PRPL: A Database Workload Specification Language, v1.3."},{"attr":"year","data_value":"1992"},{"attr":"school","data_value":"Univ. of Wisconsin-Madison"},{"attr":"author","data_value":"Tolga Yurek"},{"attr":"title","data_value":"Efficient View Maintenance at Data Warehouses."},{"attr":"year","data_value":"1997"},{"attr":"school","data_value":"University of California at Santa Barbara, Department of Computer Science"},{"attr":"editor","data_value":"Paul R. McJones"},{"attr":"title","data_value":"The 1995 SQL Reunion: People, Project, and Politics, May 29, 1995."},{"attr":"journal","data_value":"Digital System Research Center:
:
```


Figure 24: DBLP (.json)

6.4 Data retrieval

Query processing response time is evaluated based on queries with different complexity in Table 5, Table 6 and Table 7.

Table 5: Queries complexity (SigmodRecord)

Query	Query description
I	Retrieve and list all the information where the tag is "number" which is a child node of tag "issue"
II	Retrieve and list all the information where tag is "article" on condition that the value of one its child node – tag "author" is "Amihai Motro"
III	Retrieve and list all the information for all tags with name "title" where the attribute article Code is greater than "152010" and less than or equal to "152010"

Table 6: Queries complexity (NASA)

Query	Query description
I	Retrieve and list all the information where the tag is "title" which is a child node of tag "dataset"
II	Retrieve and list all the information where the tag is "other" on condition that the value of one of its child nodes - tag "lastName" is 'Jackson'
III	Retrieve and list all the information for all tags with name "other" where the value of its child node-tag "year" is greater than '1970' and less than or equal to '1990'

Table 7: Queries complexity (DBLP)

Query	Query description
I	Retrieve and list all the information where the tag is "title" which is a child node of tag "www"
II	Retrieve and list all the information where the tag is "masterthesis" on condition that the value of one of its child nodes – tag "year" is "2006"
III	Retrieve and list all the information for all tags with name "www" where the attribute key is "www/org/tpc" or the attribute mdate is "2004-12-02"

Experiment 1:
Data Filtering and Converting Response Time

In this section, data are extracted and filter from different data sources and converted into two data models: XML and JSON. Three types of datasets have been used in this experiment which are SigmodRecord, NASA and DBLB. In this experiment, data filtering response time has been calculated 4 times. The last column shows the average data filtering response time for each data model. Based on Table 8, the result of data filtering response time JSON(DAWG+) is reduced to 2% - 3% compared to others using SigmodRecord dataset. Meanwhile, based Table 9, the result of data insertion response time JSON(DAWG) is reduced to 5% - 7% compared to others using NASA dataset. Then, based Table 10, the result of data insertion response time JSON(DAWG+) is reduced to 7% - 10% compared to others using DBLP dataset. The number of percentages for each data model can be calculated based on the following formula:

$$\frac{Avg.of\ (XML) - Avg.of\ JSON}{Avg.of\ XML}\ x\ 100$$

Meanwhile, Figure 6.1, Figure 6.2 and Figure 6.3 represent response time for data insertion in line graph based on result in Table 8, Table 9 and 10.

Table 8: Data filtering response time (SigmodRecord)

	Data filtering response time (ms)				
Approach	**1st**	**2nd**	**3rd**	**4th**	**Avg.**
XML (DAWG)	435	445	450	454	446
XML (DAWG+)	420	425	423	427	424
JSON (DAWG)	390	395	398	388	393
JSON (DAWG+)	375	370	365	368	370

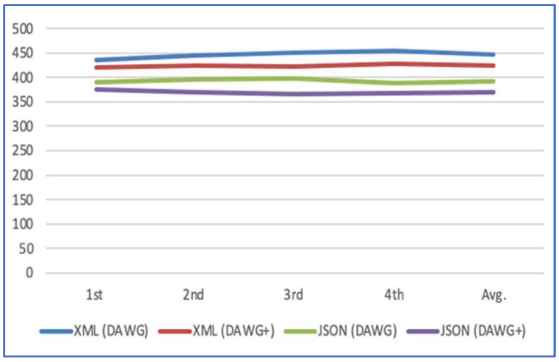

Figure 25: Data filtering response time (SigmodRecord)

Table 9: Data filtering response time (NASA)

	Data filtering response time (ms)				
Approach	**1st**	**2nd**	**3rd**	**4th**	**Avg.**
XML (DAWG)	1010	1003	1005	1008	1007
XML (DAWG+)	990	995	998	985	992
JSON (DAWG)	980	975	982	970	977
JSON (DAWG+)	950	945	958	948	950

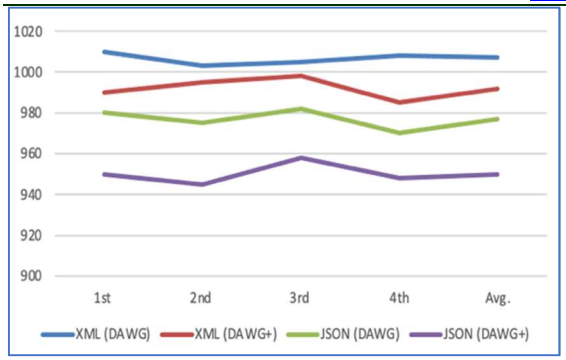

Figure 26: Data filtering response time (NASA)

Table 10: Data filtering response time (DBLP)

	Data filtering response time (ms)				
Approach	**1st**	**2nd**	**3rd**	**4th**	**Avg.**
XML (DAWG)	2130	2110	2115	2118	2118
XML (DAWG+)	2100	2090	2105	2088	2096
JSON (DAWG)	1980	1985	1990	1998	1988
JSON (DAWG+)	1750	1780	1650	1680	1715

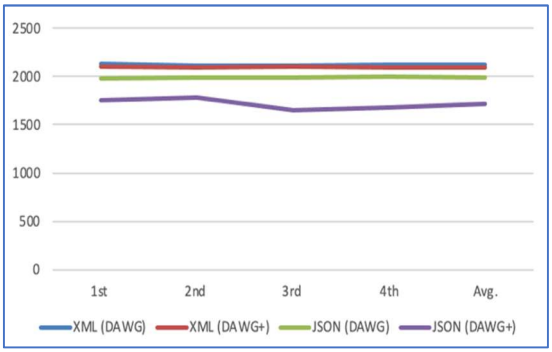

Figure 27: Data filtering response time (DBLP)

Experiment 2:
Query processing response time

In this section, we evaluated the performance of XML and JSON based on query processing response time. Three (3) different queries were executed based on specified statement in Table 5, Table 6 and Table 7, and query processing response time were executed 4 times.

Table 11 –13 show the query processing response time for SigmodRecord dataset. The result shows JSON(DAWG+) in three different queries complexity are reduce between 5% - 8% compared to others. The number of percentages for each data model can be calculated based on the following formula:

$$\frac{Avg.query\ processing\ (XML) - Av\ .query\ processing\ (JSON)}{Avg.query\ processing\ (XML)} \times 100$$

Table 11: Query processing response time – Query I

	Data filtering response time (ms)				
Approach	**1st**	**2nd**	**3rd**	**4th**	**Avg.**
XML (DAWG)	765	750	745	760	755
XML (DAWG+)	640	645	650	642	644
JSON (DAWG)	635	638	640	630	636
JSON (DAWG+)	630	633	625	630	630

Table 12: Query processing response time – Query II

	Data filtering response time (ms)				
Approach	**1st**	**2nd**	**3rd**	**4th**	**Avg.**
XML (DAWG)	780	765	775	770	773
XML (DAWG+)	770	765	760	768	766
JSON (DAWG)	750	755	758	760	756
JSON (DAWG+)	730	735	740	728	733

Table 13: Query processing response time – Query III

	Data filtering response time (ms)				
Approach	**1st**	**2nd**	**3rd**	**4th**	**Avg.**
XML (DAWG)	790	775	780	785	783
XML (DAWG+)	780	785	770	772	777
JSON (DAWG)	720	715	710	728	718
JSON (DAWG+)	715	704	695	698	703

Meanwhile, Figure 28 - 30 represent the line graph for query processing response time for SigmodRecord dataset in milliseconds (ms).

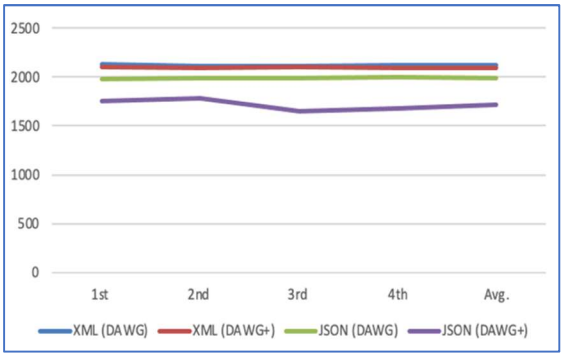

Figure 28: Query processing response time – Query I

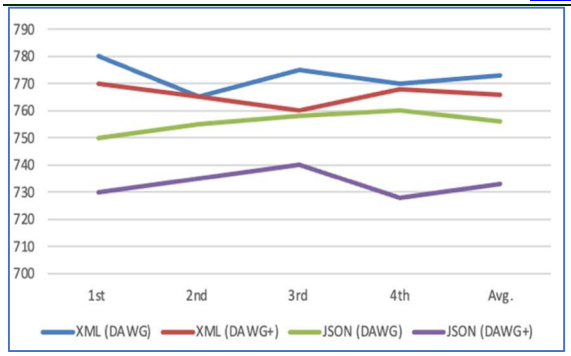

Figure 29: Query processing response time – Query II

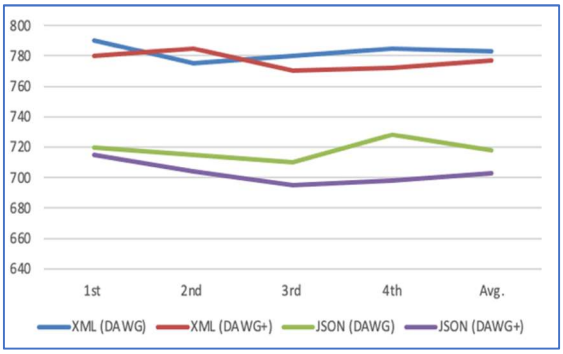

Figure 30: Query processing response time – Query III

Tables 14 – 16 show the query processing response time for NASA dataset. The result shows JSON(DAWG+) in three different queries complexity are reduce between 8% - 12% compared to others.

Table 14: Query processing response time – Query I

	Query processing response time (ms)				
Approach	**1st**	**2nd**	**3rd**	**4th**	**Avg.**
XML (DAWG)	980	974	975	970	975
XML (DAWG+)	970	965	960	972	967
JSON (DAWG)	940	945	944	938	942
JSON (DAWG+)	930	925	928	938	930

Table 15: Query processing response time – Query II

	Query processing response time (ms)				
Approach	**1st**	**2nd**	**3rd**	**4th**	**Avg.**
XML (DAWG)	1005	995	1002	998	1000
XML (DAWG+)	998	989	990	995	993
JSON (DAWG)	970	975	980	974	975
JSON (DAWG+)	950	945	948	955	950

Table 16: Query processing response time – Query III

	Query processing response time (ms)				
Approach	**1st**	**2nd**	**3rd**	**4th**	**Avg.**
XML (DAWG)	980	995	990	998	991
XML (DAWG+)	978	970	974	972	974
JSON (DAWG)	960	965	968	964	964
JSON (DAWG+)	955	958	950	948	953

Meanwhile, Figure 31 - 33 represent the line graph for query processing response time for NASA dataset in milliseconds (ms).

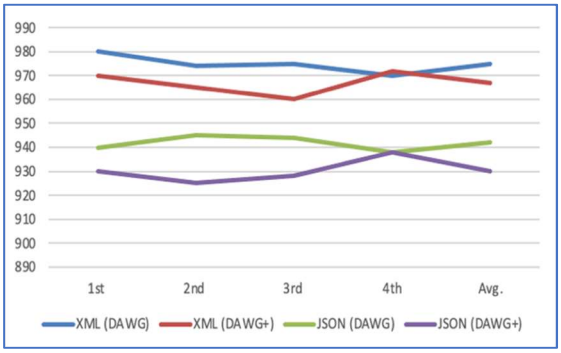

Figure 31: Query processing response time – Query I

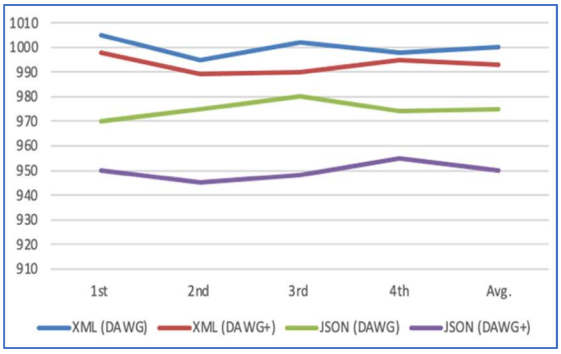

Figure 32: Query processing response time – Query II

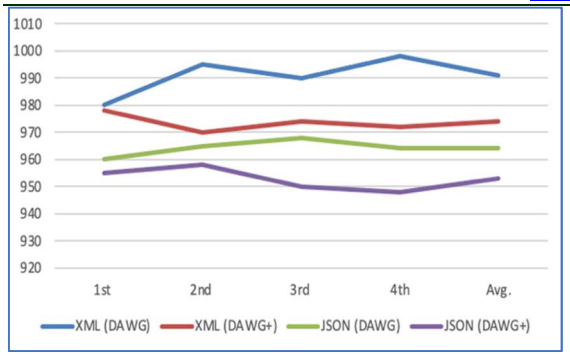

Figure 33: Query processing response time – Query III

Tables 17 –19 show the query processing response time for DBLP dataset. The result shows JSON(DAWG+) in three different queries complexity are reduce between 8% - 15% compared to others.

Table 17: Query processing response time – Query I

	Query processing response time (ms)				
Approach	**1st**	**2nd**	**3rd**	**4th**	**Avg.**
XML (DAWG)	955	958	950	948	953
XML (DAWG+)	840	844	848	852	846
JSON (DAWG)	824	820	822	825	823
JSON (DAWG+)	820	815	818	822	819

Table 18: Query processing response time – Query II

	Query processing response time (ms)				
Approach	**1st**	**2nd**	**3rd**	**4th**	**Avg.**
XML (DAWG)	725	715	730	722	723
XML (DAWG+)	712	718	720	710	715
JSON (DAWG)	710	705	708	712	709
JSON (DAWG+)	698	702	690	688	695

Table 19: Query processing response time – Query III

	Query processing response time (ms)				
Approach	**1st**	**2nd**	**3rd**	**4th**	**Avg.**
XML (DAWG)	980	975	982	978	979
XML (DAWG+)	975	970	978	982	976
JSON (DAWG)	925	920	934	938	929
JSON (DAWG+)	915	910	912	908	911

Meanwhile, Figure 34 - 36 represent the line graph for query processing response time for DBLP dataset in milliseconds (ms).

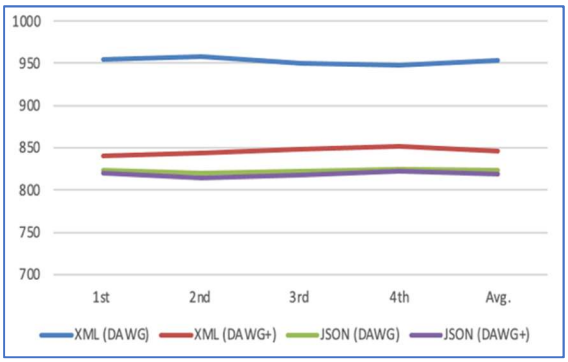

Figure 34: Query processing response time – Query I

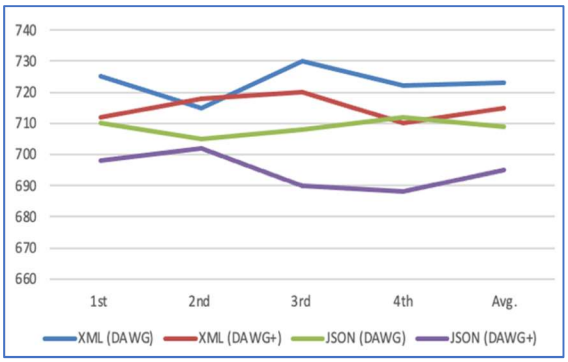

Figure 35: Query processing response time – Query II

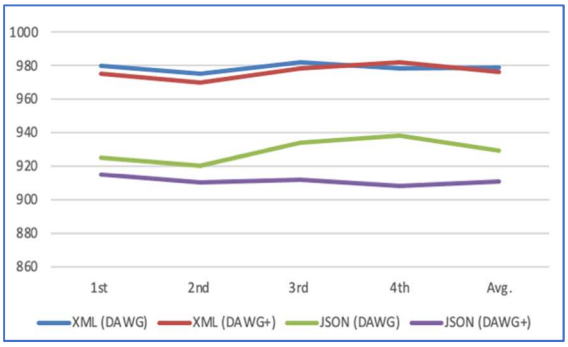

Figure 36: Query processing response time – Query III

© 2023 Little Lion Scientific
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

7. CONCLUSION AND FUTURE WORK

Three algorithms for data filtering in data integration has reviewed in this paper. Based on the previous research, DAWG is one of the best approaches for data filtering. In this research, DAWG+ has been proposed by modification of the algorithm based on DAWG. In this modification algorithm, a library table is designed to detect and remove special or unused characters from data sources. Three different data sources have been used in experiments: SigmodRecord, NASA and DBLP.

Based on the result in the experiments, DAWG+ produce better performance in term of data filtering response time and query processing response time compared to DAWG algorithm. In conclusion, by modification of DAWG algorithm, removing special or unused characters data from different data sources in data filtering process is proven to improve the efficiency in data integration.

This research will be extended in future work by modification of DAWG+ algorithm to support different format of data, different size of data (small, medium, large) and different special or unused characters.

ACKNOWLEDGMENT:

This research was supported by Ministry of Higher Education (MOHE) through Fundamental Grant Scheme (FRGS/1/2020/ICT06/UNISZA/03/2).

REFERENCES:

[1] Yang, J. L. (2020). Brief introduction of medical database and data mining technology in big data era. . *Journal of Evidence Based Medicine, 13(1), 57-69.*

[2] Balakayeva, G. T. (2019). Using NoSQL for processing unstructured big data. News of the National Academy of Sciences of the Republic of Kazakhstan, Series of Geology and Technical Sciences.

[3] Sibanda, E. L. (2020). Use of data from various sources to evaluate and improve the prevention of mother-to-child transmission of HIV programme in Zimbabwe. *a data integration exercise. Journal of the International AIDS Society, 23, e25524.*

[4] Yusof, M. K., Man, M. "Efficiency of JSON approach for Data Extraction and Query Retrieval," *The Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, 4(1), pp. 203 - 214, 2016, doi: 10.11591/ijeecs.v4.i1.pp203-214.

[5] Feng, Z. R., & Takaoka, T. (1987). On improving the average case of the Boyer-Moore string matching algorithm. *Journal of Information Processing*, *10*(3), 173-177.

[6] Prasad, R., & Agarwal, S. (2010). Parameterized string matching: An application to software maintenance. *ACM SIGSOFT Software Engineering Notes*, *35*(3), 1-5.

[7] Lockard, C. D. (2018). Ceres: Distantly supervised relation extraction from the semi-structured web. . *arXiv preprint arXiv:1804.04635.*

[8] Rosati, R. Conceptual modeling for data integration. In Conceptual Modeling: Foundations and Applications (pp. 173-197). Springer, Berlin, Heidelberg.

[9] Cima, G., Console, M., Lenzerini, M., & Poggi, A. (2021, June). Abstraction in data integration. In 2021 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS) (pp. 1-11). IEEE.

[10] Li, W., Yan, L., Zhang, F., & Chen, X. (2018). A formal approach of construction fuzzy XML data model based on OWL 2 Ontologies. *IEEE Access*, *6*, 22025-22033.

[11] Klaib, A. A., Milad, A. A., & Algaet, M. A. (2021, September). A New Approach for Labelling XML Data. In *2021 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)* (pp. 603-607). IEEE.

[12] Sabri Ahmad, I. A., & Man, M. (2018). Multiple types of semi-structured data extraction using wrapper for extraction of image using DOM (WEID). In *Regional Conference on Science, Technology and Social Sciences (RCSTSS 2016) Theoretical and Applied Sciences* (pp. 67-76). Springer Singapore.

[13] Bourhis, P., Reutter, J. L., & Vrgoč, D. (2020). JSON: Data model and query languages. *Information Systems*, *89*, 101478.

© 2023 Little Lion Scientific
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[14] Lv, T., Yan, P., & He, W. (2018, August). Survey on JSON data modelling. In *Journal of Physics: Conference Series* (Vol. 1069, No. 1, p. 012101). IOP Publishing.

[15] Breje, A. R., Gyorödi, R., Gyorödi, C., Zmaranda, D., & Pecherle, G. (2018). Comparative study of data sending methods for XML and JSON models. *International Journal of Advanced Computer Science and Applications*, *9*(12).

[16] Yusof, M. K, Man, M., Hamzah, W. A. F., Safei, S., Ismail, Ismail, "Native JSON Model for Data Integration in Business Intelligent Applications," *Journal of Theoretical and Applied Information Technology*, 100(18), pp. 5384 - 5395, 2022.

[17] Riza, L. S., Rachmat, A. B., Munir, T. H., & Nazir, S. (2019). Genomic repeat detection using the Knuth-Morris-Pratt algorithm on R high-performance-computing package. Int. J. Advance Soft Compu. Appl, 11(1), 94-111.

[18] Ojugo, A., & Eboka, A. O. (2019). Signature-based malware detection using approximate Boyer Moore string matching algorithm. *International Journal of Mathematical Sciences and Computing*, *5*(3), 49-62.

[19] Yan, N., Yafei, J., & Shenglan, Y. (2020, June). Application Research of Boyer-Moore Algorithm in Cryptography. In *2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)* (pp. 645-648). IEEE.

[20] Abu Bakar, W. A., Man, M., Man, M., Abdullah, Z. "i-Eclat: performance enhancement of eclat via incremental approach in frequent itemset mining," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, 18(1):562, doi: 10.12928/telkomnika.v18i1.13497

[21] Sharapova, E. (2020). Computational load reduction of fuzzy duplicate detection in large amounts of information. In *IOP Conference Series: Materials Science and Engineering* (Vol. 734, No. 1, p. 012119). IOP Publishing.

[22] Bisandu, D. B., Gurumdimma, N. Y., Alams, M. T., & Datiri, D. D. (2018). An enhanced text mining approach using dynamic programming.

[23] Zhang, Z. (2022). Review on String-Matching Algorithm. In *SHS Web of Conferences* (Vol. 144, p. 03018). EDP Sciences.